Personal Parallel Programming (P3) Using SHSD

Abdelrahman Osman[1], Marwan A. Al-Namari[2]
[12](Computer Science, College of Computer at Al-Qunfudah/ Umm Al-Qura University, KSA)
aamosman7
Corresponding Author: Abdelrahman Osman

Abstract: *Most of our universities have no computer laboratories for practicing parallel programming. In this paper, we will show how students can get benefits from graphic processing unit (GPU) devices by using them as P3 in solving parallel programming assignments. The main advantage of GPU computing is that it provides cheap parallel processing environments for those who need to solve Single Program Multiple Data (SPMD) problems. In this paper, Single Host Single Device (SHSD) is proposed to be used for solving many different problems taught in parallel programming course.*

Date of Submission: 16--05-2018 Date of acceptance: 31-05-2018

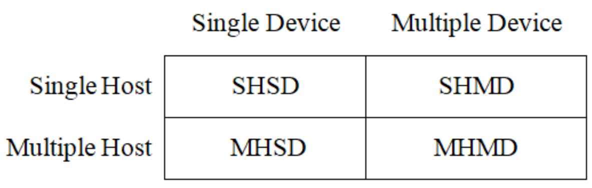
I. Introduction

GPU is a device that contains hundreds to thousands of ALUs with a same size. This makes GPU capabilities to run thousands of threads concurrently. Parallelism of threads in a GPU is suitable for executing same copy of a single program on different data [1].

GPU architecture consists of two main components, Global memory and Streaming Multiprocessors (SMs). The global memory is the main memory for the GPU and it's accessible by both GPU and CPU with high bandwidth. While SMs contains many simple cores that execute the parallel computations. Number of SMs in a device and the number of cores in SMs are differ from one device to another. For example, Fermi has 16 SMs with 32 cores on each one (with total of cores equal to 16x32=512 cores), see table 1 for different GPU devices.

II. Gpu Taxonomy

GPU computing is divided into four different classes according to the different combinations between hosts and devices. These classes are Single Host with Single Device (SHSD), Single Host with multiple Devices (SHMD), Multiple Hosts with Single Device in each (MHSD), and Multiple Hosts with Multiple Devices (MHMD), fig 1 [2].



	Single Device	Multiple Device
Single Host	SHSD	SHMD
Multiple Host	MHSD	MHMD

Fig 1: GPU Computing Taxonomy

Where SHSD is composed from one host (computer) with one GPU device installed in it. Normally the host will run the codes that are similar to conventional programming that we know (may be serial), while the parallel part of the code will be executed in the device's cores concurrently (Massive parallelism).

The processing flow of SHSD computing includes: transfer input data from CPU's memory to GPU's memory, load GPU program and execute it, and transfer results from GPU's memory to CPU's memory [3].

In SHMD one computer can host more than one GPU, and can be used to run parallel tasks in installed GPUs, with each GPU run sub-tasks in their cores.

In MHSD many computers linked together as a cluster with each node has one GPU installed in it. i.e cluster of SHSDs.

The last class is MHMD which is consists of cluster of SHMD nodes (where all nodes may have the same number of devices, or may have not).

III. Using Shsd As P3

Most of personal computers today are SHSD, which is consist of one host (computer) with one GPU device installed in. Normally the host will run the codes that are similar to conventional programming that we know (may be serial), while the parallel part of the code will be executed in the device's cores concurrently (Massive parallelism).

The processing flow of SHSD computing includes:

• Transfer input data from CPU's memory to GPU's memory.

• Load GPU program and execute it.

• Transfer results from GPU's memory to CPU's memory [3].

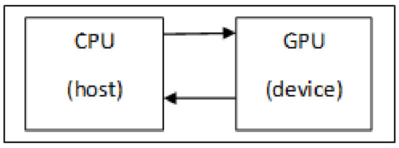


Fig **2: Shsd**

In the SHSD example in fig 3, data transferred from CPU host to GPU device are coming through a communication bus connecting GPU to CPU. This communication bus is of type PCI-Express with data transfer rate equal to 8 GB/Sec, which is the weakest link in the connection (new fast generation is available). The other links in the figure are the memory bandwidth between main memory DDR3 and CPU (42 GB/Sec), and the memory bandwidth between GPU and its global memory GDDR5 (288 GB/Sec).

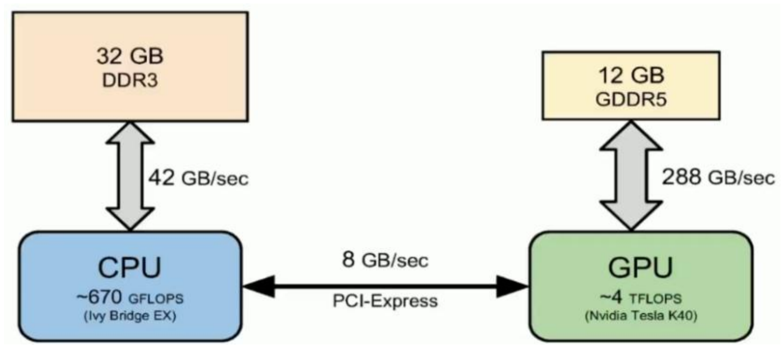


Fig 3: Example of SHSD class, from [4].

Bandwidth limited

Because transfer data between the CPU and GPU is expensive, we will always try to minimize the data transfer between the CPU and GPU. Therefore, if a processor's request data at too high a rate, the memory system cannot keep up. No amount of latency hiding helps this. Overcoming bandwidth limits are a common challenge for GPU-compute application developers [5].

IV. Parallel Programming models

In the practical side, we gave students programming assignments for shared and distributed memory models. In the shared memory model we used multithreading (Java threads, threads, and OpenMP), while in the distributed model we used MPI [7][8].

The proposed system that we need to use instead of these models is SHSD GPU computing, which is available in most of personal computers.

Our main purpose in writing parallel programs usually is to increase performance. Students should write a parallel program for different applications and make comparisons between different data sizes to test speedup and efficiency using shared and distributed memory model.

The problem faces students is the lack of labs for practicing distributed memory model, and in the shared memory model they used multicore systems which gives weak performance.

Nowadays, most of laptops, host a GPU device and this motivate us to use GPU Computing system instead of multicore system. The reason is that CPUs are low latency low throughput processors (faster for serial processing), while GPUs are high latency, high throughput processors (optimized for maximum throughput and for scalable parallel processing).

Compute Unified Device Architecture (Cuda)

There are many tools available for GPU computing programming. In this section we will talk about CUDA as a proposed tool that can be used in SHSD system.

CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables increases the computing performance by harnessing the power of the GPU devices. CUDA Split a problem into serial sections and parallel sections, Serial sections execute on the CPU as host code; Parallel sections execute on the GPU by launching a kernel, fig 4.

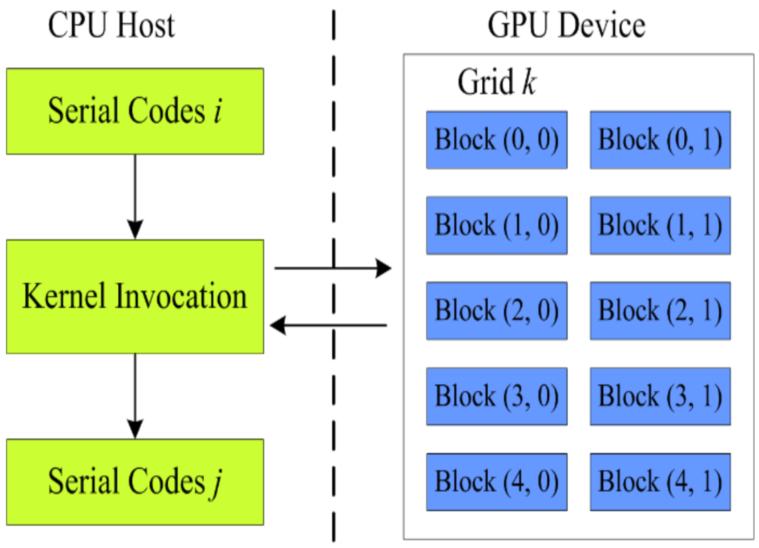


Fig 4: Using CUDA in SHSD.

V. Cuda In Shsd As P3

CUDA can run in SHSD as P3 using the following sequence of operations [6]:

- Declare and allocate host and device memory.
- Initialize host data.
- CPU transfer input data to GPU.
- Launch kernel on GPU to process the data in parallel.
- Copies results back from the GPU to the CPU.

Kernel code: the instructions actually executed on the GPU.
The following code demonstrates a portion of code that can be run in SHSD.

```
// Device code
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C)
{ int i = threadIdx.x;C[i] = A[i] + B[i];}

// Host code invoking Device code ( kernel)
int main() {
... // Kernel invocation with N threads (invoke device code from host code)
VecAdd<<<1, N>>>(A, B, C);...}
```

VI. Conclusion

Most of our universities have no dedicated labs for practicing parallel programming. Instead student used their personal laptops to do shared memory programming assignments. In this paper, we proposed SHSD GPU computing for doing the practical part of parallel programming course to achieve significant performance improvements.

References

[1]. Rege, A. Modern GPU Architecture - NVIDIA. [Cited 2016 31 Dec.].

[2]. Osman, Abdelrahman Ahmed Mohamed. "GPU Computing Taxonomy." Recent Progress in Parallel and Distributed Computing. InTech, 2017.

[3]. Harris, M. Tesla GPU Computing, A Revolution in High Performance Computing. 2009 [cited 2017 3 Jan.]; Available from: http://www.lsr.nectec.or.th/images/f/f2/Overview.pdf.
[4]. Kardos, J. Efficient Data Transfer, Advanced Aspects of CUDA. 2015 [cited 2017 3 Jan.]; Available from: http://www.youtube.com/watch?v=Yv4thF9tvPo.
[5]. Rosenberg, O. Introduction to GPU Architecture [cited 2017 17 Jan.]; Available from: http://haifux.org/lectures/267/Introduction-to-GPUs.pdf.
[6]. Harris, M. An Easy Introduction to CUDA C and C++. 2012 [cited 2016 28 Dec.]; Available from: https://devblogs.nvidia.com/parallelforall/easy-introduction-cuda-c-and-c/.
[7]. Pacheco, Peter. An introduction to parallel programming. Elsevier, 2011.
[8]. Osman, Abdelrahman. (2011). Parallel Programming, https://www.researchgate.net/publication/307955739_Parallel_Programming

Abdelrahman Osman "Personal Parallel Programming (P3) Using SHSD." IOSR Journal of Environmental Science, Toxicology and Food Technology (IOSR-JESTFT) 12.5 (2018): 42-45